Nicht für den Prüfling bestimmt

I Erläuterungen

Voraussetzungen gemäß KCBG und Abiturerlassen BG jeweils in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Kompetenzen für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzen in engem Bezug zueinander stehen. Die Operationalisierung des Bezugs zu den Kompetenzbereichen des Standardbezugs erfolgt in Abschnitt II.

Aufgabe	**Kompetenzen**				
	K1	**K2**	**K3**	**K4**	**K5**
1.1	X	X		X	
1.2	X	X			
1.3.1				X	
1.3.2				X	
1.3.3				X	
1.4	X		X		
2.1	X				X
2.2		X		X	
2.3	X		X		
2.4	X		X		
2.5.1		X		X	
2.5.2	X		X		

Inhaltlicher Bezug
Die nachfolgend ausgewiesenen Themenfelder sind die wesentliche inhaltliche Grundlage für die vorliegenden Aufgaben. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Themenfelder für die Bearbeitung nachrangig bedeutsam sein.
Q1: Objektorientierte Softwareentwicklung
Q2: Datenbanksysteme
Q3: Datenkommunikation
verbindliche Themenfelder: Objektorientierte Modellierung (Q1.1), Implementierung von Klassen und Assoziationen (Q1.2), Konzeptionelle und logische Modellierung einer Datenbank (Q2.1), Datenabfrage und Datenmanipulation mit SQL (Q2.2), Serielle Kommunikation (Q3.1)

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.1	beschreiben, erklären Entitätstypen (grafische Darstellung als Rechteck) kategorisieren gleichartige Entitäten, die sich durch die gleichen Eigenschaften (Attribute) beschreiben lassen. Im vorliegenden Modell ist beispielsweise `Kunde` ein Entitätstyp. Attribute charakterisieren einen Entitätstyp bzw. einen Beziehungstyp. Die Attribute des Entitätstyps `Kunde` sind zum Beispiel `name` und `telefon`. Die Darstellung von Attributen im ERM sind Ellipsen. Identifizierende Attribute, so genannte Schlüsselattribute, dienen der eindeutigen Identifizierung von Entitäten eines Entitätstyps. Ein Primärschlüssel kann aus mehreren identifizierenden Attributen zusammengesetzt sein und wird durch Unterstreichung gekennzeichnet. Der Primärschlüssel von Kunde ist `kNr`. Die Rauten stehen für Beziehungstypen. Durch Beziehungen (Relationships) werden die Abhängigkeiten zwischen Entitäten ausgedrückt. Ein Beziehungstyp ist die Abstraktion gleichartiger Beziehungen und kann ebenfalls Eigenschaften besitzen. Der Beziehungstyp `enthält` besitzt das Attribut `anzahl`. Über die Kardinalität wird festgelegt, wie viele Entitäten einer Entitätsmenge mit Entitäten einer anderen Entitätsmenge in Beziehung stehen können oder müssen. Die Kardinalitäten im vorliegenden ERM sind in [min,max]-Notation dargestellt und bedeuten am Beispiel des Beziehungstyps `enthält`: Jede Bestellung enthält einen (1) bis beliebig viele (m) Artikel (muss-Beziehung). Jeder Artikel kann in keiner (0) bis beliebig vielen (n) Bestellungen enthalten sein (kann-Beziehung). Die `is-a`-Beziehung (als Dreieck dargestellt) steht für Generalisierung bzw. Spezialisierung. Das Prinzip der Generalisierung wird eingesetzt, um eine übersichtliche und natürliche Strukturierung der Entitätstypen zu erreichen. Gemeinsame Eigenschaften sind dem Obertyp zugeordnet. Die Untertypen enthalten die speziellen Eigenschaften, die nicht allen Untertypen gemeinsam sind. Das vorliegende Modell zeigt die Spezialisierung von Artikeln. Der Obertyp Artikel enthält die allgemeinen Eigenschaften `artNr`, `preis` und `bestand`. Der Untertyp Lebewesen zeichnet sich zusätzlich durch die Attribute `name` und `herkunft`, der Untertyp Zubehör durch `bezeichnung` und `beschreibung` aus. beschreiben erklären	3	2	1

1.2	überführen
	`Artikel(`<u>`artNr`</u>`, preis, bestand)`
	`Lebewesen(`<u>`artNr#`</u>`, name, herkunft)`
	`Pflanzenart(`<u>`artNr#`</u>`, wuchshoehe, lichtbedarf)`
	`Fischart(`<u>`artNr#`</u>`, laenge, gruppengroesse)`
	`Zubehoer(`<u>`artNr#`</u>`, bezeichnung, beschreibung)`
	`Kunde(`<u>`kNr`</u>`, vname, nname, plz, ort, strasse, telefon)`
	`Bestellung(`<u>`bestellNr`</u>`, bestellDatum, lieferDatum, kNr#)`
	`Position(`<u>`bestellNr#`</u>`, `<u>`artNr#`</u>`, anzahl)`
	begründen
	Alle Entitätstypen werden mit ihren Attributen in Relationen überführt. Der Name des Kunden und die Adresse müssen zur Einhaltung der ersten Normalform aufgespalten werden. Die 1:n-Beziehung `gibt_auf` zwischen `Kunde` und `Bestellung` erfordert, dass der PK der 1-Relation (`kNr`) als FK in die n-Relation aufgenommen wird. n:m-Beziehungen werden durch eigene Relationen realisiert, hier am Beispiel der Beziehung `enthält`. Die PK der beteiligten Tabellen werden FK in der Beziehungsmengenrelation `Position`.
	Bei der Generalisierung wurde für jeden Entitätstyp eine Relation mit den relevanten Attributen angelegt. Durch die 1:c-Beziehung wird dem Untertyp der
		6		

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td></td><td>PK des Obertyps als FK hinzugefügt und kann hier außerdem als PK dienen. Das hat den Vorteil, dass alle Artikel der Hierarchie eine eindeutige Artikelnummer besitzen.
Die Nicht-Schlüsselattribute der Relationen sind vollständig vom jeweiligen PK abhängig und es existieren keine transitiven Abhängigkeiten, so dass die 3. Normalform eingehalten wird.</td><td></td><td>3</td><td></td></tr>
<tr><td>1.3.1</td><td>entwickeln<pre>
INSERT INTO Kunde
VALUES(7654, 'Hans', 'Hecht', '89075', 'Ulm', 'Donautor 1',
       '0177 1234567');

INSERT INTO Bestellung(bestellDatum, kNr)
VALUES('2022-05-02', 7654);

INSERT INTO Position
VALUES((SELECT bestellNr FROM Bestellung
     WHERE bestellDatum = '2022-05-02'
     AND kNr = 7654), 277, 1);

UPDATE Artikel SET bestand = bestand - 1
WHERE artNr = 277;
</pre></td><td></td><td>5</td><td>2</td></tr>
<tr><td>1.3.2</td><td>formulieren<pre>
SELECT bestellNr, k.*
FROM Bestellung b JOIN Kunde k ON(b.kNr=k.kNr)
WHERE b.lieferDatum IS null
AND k.plz LIKE '12%';
</pre></td><td></td><td>3</td><td></td></tr>
<tr><td>1.3.3</td><td>implementieren<pre>
SELECT l.herkunft, SUM(a.preis * p.anzahl) AS Gesamtumsatz
FROM Bestellung b JOIN Position p
     ON(b.bestellNr = p.bestellNr)
     JOIN Artikel a ON(p.artNr = a.artNr)
     JOIN Lebewesen l ON(a.artNr = l.artNr)
     JOIN Fischart f ON(l.artNr = f.artNr)
WHERE b.bestellDatum BETWEEN '2022-01-01' AND '2022-03-31'
GROUP BY l.herkunft;
</pre></td><td></td><td>2</td><td>3</td></tr>
</table>

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>1.4</td><td>entwickeln, zeichnen

Hinweis: Eine Darstellung ohne schwachen Entitätstyp `Position` ist als gleichwertig zu betrachten.

entwickeln
zeichnen</td><td>
3</td><td>3</td><td>4</td></tr>
<tr><td></td><td>**Summe 40**</td><td>**12**</td><td>**18**</td><td>**10**</td></tr>
</table>

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.1	beschreiben, erläutern Das Prinzip der Datenkapselung stellt sicher, dass nur ein kontrollierter Zugriff von außen auf Attribute und Methoden einer Klasse möglich ist. Mithilfe des Zugriffsmodifikators `private` (-) wird die Sichtbarkeit eines Attributs so eingeschränkt, dass nur innerhalb der Klasse darauf zugegriffen werden kann. Ein Zugriff von anderen Klassen ist dann nur über definierte Schnittstellen möglich, beispielsweise über get- und set-Methoden. Durch den Zugriffsmodifizierer `public` (+) ist ein Zugriff von außerhalb und innerhalb der Klasse möglich. Die Zugriffsart `protected` (#) in der Klasse Lebewesen ermöglicht den Zugriff innerhalb der Klasse und seiner Spezialisierungen (z.B. aus der Subklasse `Fischart`). In der Aquaristik-Software muss sichergestellt werden, dass die Attribute vor missbräuchlichen oder versehentlichen Veränderungen geschützt werden. Der Container `lebewesen` der Klasse `Aquarium` soll nur mithilfe der öffentlichen Methode `besetzeAquarium()` verändert werden können. Somit wird verhindert, dass durch direktes Hinzufügen von Fischen in ein Aquarium ein Überbesatz entstehen kann. beschreiben erläutern	 3	 1	
2.2	entwickeln, zeichnen : Fischart name = Diskus Red spotted green herkunft = Südamerika minTemp = 28 maxTemp = 30 laengeInCM = 23 gruppengroesse = 8 : Aquarium nr = 18 laenge = 160 breite = 60 hoehe = 60 nettoVolumenInLiter = 504 literBelegt = 368 anzahlFischArten = 1 : Pflanzenart name = Amazonas Schwertpflanze herkunft = Südamerika minTemp = 22 maxTemp = 28 wuchshoehe = 70 lichtbedarf = mittel : Pflanzenart name = Zwergpfeilkraut herkunft = Südamerika minTemp = 20 maxTemp = 28 wuchshoehe = 10 lichtbedarf = mittel : Aquarium nr = 13 laenge = 100 breite = 40 hoehe = 40 nettoVolumenInLiter = 140 literBelegt = 140 anzahlFischArten = 2 : Fischart name = Panda-Panzerwels herkunft = Südamerika minTemp = 23 maxTemp = 26 laengeInCM = 5 gruppengroesse = 5 vertraeglichMit vertraeglichMit : Fischart name = Blauer Neon herkunft = Südamerika minTemp = 24 maxTemp = 32 laengeInCM = 3 gruppengroesse = 15 entwickeln zeichnen	 4	4	

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.3	überführen, implementieren			

```
public class Aquarium {
  private int nr;
  private int laenge;
  private int breite;
  private int hoehe;
  private int nettoVolumenInLiter;
  private int literBelegt;
  private int anzahlFischArten;
  private static int autowert = 1;
  private List<Lebewesen> lebewesen;

  public Aquarium(int laenge, int breite, int hoehe) {
    this.nr = autowert++;
    this.laenge = laenge;
    this.breite = breite;
    this.hoehe = hoehe;
    this.nettoVolumenInLiter =
              (int)((laenge*breite*hoehe/1000) * 0.875);
    this.literBelegt = 0;
    this.anzahlFischArten = 0;
    this.lebewesen = new List<>();
  }

  public String ermittleHerkunft() {
    String s = "";
      if(lebewesen.size() > 0)
        s = lebewesen.get(0).herkunft;
    return s;
  }

  public boolean pruefeFreiesVolumen(Fischart art) {
    int literBedarf = art.getLaengeInCM() *
                      art.getGruppengroesse() * 2;
    if(literBedarf + literBelegt <= nettoVolumenInLiter)
      return true;
    return false;
  }

  public boolean besetzeAquarium(Lebewesen art) {
    boolean ok = true;
    if(art instanceof Fischart) {
      Fischart f = (Fischart) art;
      if(pruefeFreiesVolumen(f)) {
        lebewesen.add(art);
        anzahlFischArten++;
        literBelegt += f.getLaengeInCM() *
                       f.getGruppengroesse() * 2;
      } else {
        ok = false;
      }
    } else {
      lebewesen.add(art);
    }
    return ok;
  }
}
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III

```
public abstract class Lebewesen {
  protected String name;
  protected String herkunft;
  protected int minTemp;
  protected int maxTemp;

  public Lebewesen(String name, String herkunft, int
                   minTemp, int maxTemp) {
    this.name = name;
    this.herkunft = herkunft;
    this.minTemp = minTemp;
    this.maxTemp = maxTemp;
  }
}

public class Fischart extends Lebewesen{
  private int laengeInCM;
  private int gruppengroesse;
  private List<Fischart> vertraeglichMit;

  public Fischart(String name, String herkunft, int minTemp,
               int maxTemp, int cm, int gruppengroesse) {
    super(name, herkunft, minTemp, maxTemp);
    this.laengeInCM = cm;
    this.gruppengroesse = gruppengroess;
    vertraeglichMit = new List<>();
}

  public void hinzufuegenVertraeglichMit(Fischart art) {
    this.vertraeglichMit.add(art);
  }
}
```

Aufg.	erwartete Leistungen	I	II	III
	überführen	6		
	implementieren		6	4

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.4	entwickeln, zeichnen			

findeAquarium(art: Fischart)

```
gesuchtesAquarium := null
leeresAquarium := null
für jedes Aquarium a aus aquarien
    ok := false
    genugPlatz := a.pruefeFreiesVolumen(art)
    leeresAquarium = null und
    a.anzahlFischArten = 0 und genugPlatz ?
    J:
        leeresAquarium := a
    N:
        herkunft := a.ermittleHerkunft()
        herkunft = art.herkunft ?
        J:
            ok := true
            a.anzahlFischArten <> 0 und
            art.vertraeglichMit.anzahl <> 0 ?
            J:
                für jedes Lebewesen l aus a.lebewesen
                    l ist Fischart-Objekt und
                    (l ist unverträglich mit art
                    oder l = art) ?
                    J:
                        ok := false
                        Abbruch
                    N:
                        ø
            N:
                ø
            ok und genugPlatz ?
            J:
                gesuchtesAquarium := a
                Abbruch
            N:
                ø
        N:
            ø
gesuchtesAquarium = null ?
J:
    gesuchtesAquarium := leeresAquarium
N:
    ø
Rückgabe: gesuchtesAquarium
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III
	entwickeln		4	6
	zeichnen	2		

Aufg.	erwartete Leistungen	BE I	BE II	BE III

2.5.1	implementieren
	`public class Steuerung {`
	`  private Aquarium aqua;`
	`  private float optimaleTemp;`
	`  private VUSB vusb;`
	
	`  public Steuerung(Aquarium aqua, int device) {`
	`    this.aqua = aqua;`
	`    vusb = new VUSB(device);`
	`    optimaleTemp = ermittleOptimaleTemp();`
	`  }`
	
	`  public void run() {`
	`    vusb.writeDigital(128);`
	`    boolean heizstabIsEingeschaltet = true;`
	`    while(ermittleAktuelleTemp() < optimaleTemp)`
	`      warteMin(1);`
	`    vusb.writeDigital(0);`
	`    heizstabIsEingeschaltet = false;`
	`    int counter = 0;`
	`    int wartezeit = 5;`
	`    while(true) {`
	`      warteMin(wartezeit);`
	`      float tempAktuell = ermittleAktuelleTemp();`
	`      if(tempAktuell < (optimaleTemp - 0.5f)) {`
	`        if(!heizstabIsEingeschaltet()) {`
	`          vusb.writeDigital(128);`
	`          heizstabIsEingeschaltet = true;`
	`        }`
	`        wartezeit = 1;`
	`        counter++;`
	`      } else if(tempAktuell > (optimaleTemp + 0.5f)) {`
	`        if(heizstabIsEingeschaltet) {`
	`          vusb.writeDigital(0);`
	`          heizstabIsEingeschaltet = false;`
	`        }`
	`        wartezeit = 1;`
	`        counter++;`
	`      } else {`
	`        if(heizstabIsEingeschaltet) {`
	`          vusb.writeDigital(0);`
	`          heizstabIsEingeschaltet = false;`
	`        }`
	`        counter = 0;`
	`        wartezeit = 5;`
	`      }`
	`      if(counter >= 5) {`
	`        if(heizstabIsEingeschaltet)`
	`          vusb.writeDigital(129);`
	`        else`
	`          vusb.writeDigital(1);`
	`      }`
	`    }`
	`  }`
		2	5	5

Aufg.	erwartete Leistungen	BE I	BE II	BE III
	(Fortsetzung Code, siehe unten)			

```
  private float ermittleAktuelleTemp() {
    float resultTemp = 0f;
    for(int i = 0; i < 20; i++) {
      int analogValue = vusb.readAnalog1();
      float temp = (5f*100f*analogValue)/1024f;
      resultTemp += temp;
      warteMs(10);
    }
    return resultTemp/20;
  }
}
```

Aufg.	erwartete Leistungen	I	II	III
2.5.2	entwickeln		4	4

		I	II	III
	Summe 60	**17**	**24**	**19**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Bei der Bewertung und Beurteilung der Übersetzungsleistung in den Fächern Latein und Altgriechisch sind die Bestimmungen des § 9 Abs. 14 OAVO in Verbindung mit Anlage 9c anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO sowie Anlage 9c zu § 9 Abs. 14 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt.

Für die Bewertung in den modernen Fremdsprachen ist der „Erlass zur Bewertung und Beurteilung von schriftlichen Arbeiten in allen Grund- und Leistungskursen der neu beginnenden und fortgeführten modernen Fremdsprachen in der gymnasialen Oberstufe, dem beruflichen Gymnasium, dem Abendgymnasium und dem Hessenkolleg“ vom 7. August 2020 (ABl. S. 519) zugrunde zu legen. Demnach erfolgt die Bewertung und Beurteilung mit der Maßgabe, dass lediglich bei der Ermittlung des Prüfungsergebnisses (Note) aus Prüfungsteil 1 und 2 gerundet wird.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Als Kriterien für die Bewertung und Beurteilung dienen unter Beachtung der Zielsetzung der gymnasialen Oberstufe nach § 1 Abs. 2 OAVO neben dem Inhaltlichen auch die in den Kerncurricula genannten überfachlichen Kompetenzen, insbesondere die Sprachkompetenz und Wissenschaftspropädeutik; dies zeigt sich u.a. in qualitativen Merkmalen wie Strukturierung, Differenziertheit, (fach-)sprachlicher Gestaltung und Schlüssigkeit der Argumentation.

Im Fach Praktische Informatik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45% der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75% der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	Bewertungseinheiten in den Anforderungsbereichen			Summe
	AFB I	AFB II	AFB III	
1	12	18	10	**40**
2	17	24	19	**60**
Summe	**29**	**42**	**29**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.